Perl Tools for Automating Satellite Ground Systems

David McLean, Therese Haar, and James McDonald

Honeywell
7515 Mission Drive
Lanham, Md. 20706

Funded under contract with NASA / GSFC / Information Systems Center

Abstract

The freeware scripting language Perl offers many opportunities for automating satellite ground systems for new satellites as well as older, in situ systems. This paper describes a toolkit that has evolved from of the experiences gained by using Perl to automate the ground system for the Compton Gamma Ray Observatory (CGRO) and for automating some of the elements in the Earth Observing System Data and Operations System (EDOS) ground system at Goddard Space Flight Center (GSFC). CGRO is an older ground system that was forced to automate because of fund cuts. Three 8 hour shifts were cut back to one 8 hour shift, 7 days per week. EDOS supports a new mission called Terra, launched December 1999 that requires distribution and tracking of mission-critical reports throughout the world. Both of these ground systems use Perl scripts to process data and display it on the Internet as well as scripts to coordinate many of the other systems that make these ground systems work as a coherent whole. Another task called Automated Multimodal Trend Analysis System (AMTAS) is looking at technology for isolation and recovery of spacecraft problems. This effort has led to prototypes that seek to evaluate various tools and technology that meet at least some of the AMTAS goals. The tools, experiences, and lessons learned by implementing these systems are described here.

Compton Gamma Ray Observatory Ground System Automation

CGRO was launched on April 5 1991 aboard the Space Shuttle Atlantis. Its principal mission is to study the gamma ray sources throughout the universe. After the primary mission was accomplished the spacecraft was reboosted into a higher orbit to extend its mission life by at least another 10 years. However, funding cuts required the ground system staff be cut from three 8-hour shifts to one 8-hour shift so ground system automation was an absolute requirement.

Many of the Perl DataTools scripts have evolved from the experience of looking for automation tools [1] and automating the CGRO ground system, now known as ROBOTT (Reduced Operations By Optimizing Tasks and Technologies) [2]. ROBOTT uses Perl scripts that coordinate the sequences of activities required to support real-time Tracking and Data Relay Satellite System (TDRSS) activities for CGRO, called pass activities, as well as pre and post pass activities. Schedule automation is very important to CGRO because all of its data must be down-linked in real-time and therefore it requires all the TDRSS contact time that is possible. These pre pass activities start by executing Perl scripts that generate a first cut TDRSS schedule for CGRO. Then, once a day, scripts download the confirmed TDRSS schedule from the Command Management System. Next, a script pulls time and pass parameter information from the confirmed TDRSS schedule and creates a CRON table to launch a day's worth of pass activities at the appropriate times. The CRON table must be set up with start times adjusted by 2 minutes before the actual TDRSS contact so the API in Perl module DateCalc.pm was used to make these adjustments. When the CRON scripts are launched, they start software that monitors and controls activities during the pass. The monitor and control software are GSFC developed tools known as the Generic Spacecraft

Analyst Assistant (GenSAA) [3] and the Generic Inferential Executor (Genie) [4]. When anomalies are identified during pass operations, Perl scripts send pages through a firewall to a ROLM phone system to alert system engineers. At the end of each pass engineering data is also sent through the firewall to a WWW site where it is available for engineers and scientists. Other Perl scripts generate the HTML that loads a graphic timeline applet that displays a day's worth of TDRSS passes. Perl scripts also generate some HTML that documents the scripts that make up ROBOTT.

The ROBOTT experience has inspired the Perl DataTools effort by making us consider what tools would really be more appropriate if we had to do it all over again. The data moving tools and the numeric paging scripts are truly portable and can be used for any application with similar needs. However there are many other mission specific scripts that could have been made more flexible and extensible by the use of a toolkit that is more database like. The Perl DataTools counting and database scripts were created with this kind of flexibility in mind and should make future ground system automation easier to maintain and to port to similar applications.

The GenSAA/Genie tools are written in C++ and use the CLIPS rule-based system to represent constraint knowledge. This diversity of languages requires the maintainer of the complete ground system to know all these languages and technology to modify the system in various ways. It would be much simpler to keep as much of the code as possible in Perl because it is a much higher level language than C++. We estimate the code ratio for Perl vs. C++ is about 1 to 10 to do similar things. Also, CLIPS is a huge system and rule firing is difficult to control -- There must be rules to control rule firing. Our experience has shown that a simple backward chaining system achieves the same result and is much easier to maintain. This has been the motivation behind the Perl Inference Engine (PIE) and A Plan Executor (APE) [5].

The Earth Science Enterprise Application

NASA's Earth Observing System (EOS), formerly known as NASA's Mission to Planet Earth, objective is to study the Earth via a series of polar-orbiting and low inclination satellites over the next two decades. The data systems supporting the capture and processing of telemetry, from a collection of EOS satellites, is supported through NASA's EOS Data Information System (EOSDIS). The EOS Data and Operations System (EDOS) element is responsible for the data capture and initial processing of the raw telemetry known as level zero processing as well as the transfer of command data blocks to the spacecraft.

The first spacecraft to be supported by the Earth Science Ground System is an international satellite named Terra, formerly referred to by the name AM-1. The Terra satellite was sent into a polar orbit via an Atlas IIAS launch vehicle, from Vandenberg AFB, California December 1999. The EOS Terra Spacecraft is a joint project between the United States, Japan, and Canada. There are a total of five instruments on the Terra Spacecraft, three provided by the United States, one from Japan, and one from Canada. The Terra Satellite telemetry is based on the Consultative Committee for Space Data Systems recommended format. Each instrument has a defined group of unique Application Process Identification (APID) numbers that define each instrument team of CCSDS telemetry packets. EDOS receives and processes the raw telemetry simultaneously on three return link channels and transfers the forward link command data to the Terra Spacecraft. Two return links support the low rate data, one kbps to 512 kbps, and one high rate return link supports 150 mbps.

Perl Report Generators for the Earth Science Enterprise

Access to telemetry status is critical to the internationally based instrument teams and the EOSDIS ground system. End-to-end ground system testing in support of pre-launch, launch, and post-launch activities increasingly demonstrated a need to provide access to data processed and captured by EDOS. Selective data points and files within EDOS are transferred to a remote server for processing and display. To meet the needs of the international cross-functional team, the concept of using WWW-based technology to facilitate automated dynamic reports was proposed. The selection of tools, such as Perl and the Apache server software, was based on their proven capabilities for automation of the CGRO ground system. Some of the valuable information provided by these Perl-generated reports are:

- Real-Time Data Files by APID including Spacecraft Start and Stop Time.
- Expedited Data Files by APID including Spacecraft Start and Stop Time.
- Product Data Sets Delivery Files including Spacecraft Start and Stop Time.
- Real-Time Data File Customer Delivery Report including product delivery completion time.
- Expedited Data Sets Product Customer Delivery Report including product delivery completion time.
- A Product Data Sets Customer Delivery Report including delivery completion time.
- Data Product Quality Report including telemetry gap information.
- Spacecraft Contact Summary Report.
- Performance Summary Report

These dynamic reports have provided an invaluable method for tracking telemetry through the production processing stages of EDOS. For example, it provides many EOSDIS team members a method of verifying receipt of telemetry into rate buffer files, production data processing status, and data quality. The data quality information provided in the EDOS detail production data sets provides the ground system with an immediate view of satellite data quality captured and processed. This has saved valuable time in defining and addressing unexpected quality data results by management, flight operations, the spacecraft team, EDOS, the instrument teams, and the Distributed Active Archive Center. In addition, the EOSDIS ground system has benefited from cost savings due to the rapid prototyping and implementation of the Perl generated accounting reports. The cost saving was realized through the removal of EOSDIS accounting requirements on EDOS and the System Monitoring and Coordination Center because data accountability for the Distributed Active Archive Center is centered on this WWW-based solution.

EDOS Database Access Using Perl

Unlike the CGRO ground system, EDOS has not yet evolved into a fully automated system. However, as EDOS moves toward this goal through the automation of data access and publishing. EDOS stores data and status information in an Oracle database that is protected by a firewall. Perl automation scripts were written to access the desired information and put it into the database where the information can be stored and manipulated prior to its presentation to managers and other customers through a WWW interface. Perl provides an excellent API to databases known as DBI. This interface was used to accept SQL queries embedded within Perl scripts to access data in the database. Direct access of the data by the script offers many advantages because the data can be formatted and massaged by the script (a task that Perl is very well suited to) before it is put into the database or displayed on a report. The database update scripts are executed automatically through CRON and the display scripts are executed as a Common Gateway Interface (CGI) in response to WWW form processing. The bottom line is that automation of the update and display of database information on the WWW can be achieved easily, efficiently, and inexpensively using Perl.

Near Real-Time EDOS Displays

A major goal of EDOS management is to allow remote access to statistical information that is presented in a concise form. The power of Perl to access, search, and manipulate data leads to simple and easy to maintain report generators, including HTML generators. Perl modules such as GIFgraph.pm, a plotting module, allow easy display of graphical data the WWW. A near real-time status display was prototyped to provide mission critical statistics on forward and return link processing. The parameters are displayed in both plain text and graphically. Some of the critical information contained in these displays include CCSDS data units received, the frame error, flywheel error, correctable virtual channel data unit, uncorrectable virtual channel data unit, symbols corrected using Reed-Solomon, command data blocks received, and out of sequence command data blocks. WWW display of data such as this frees computing resources in the control room for more critical real-time monitoring.

Automated Multimodal Trend Analysis System (AMTAS) Prototypes

AMTAS [6] research has led to a number of Perl prototypes that allow experimentation with univariate and multivariate technologies for detecting trends in data. The univariate techniques use a “sliding window” (queue) of time series data that represents the current “average” state of the data and a nominal “model” that represents the expected state of the data. Goodness of fit tests are used to detect unexpected trends in nominal (chi square test) and ordinal (Kolmogorov-Smirnov test) data types. Simple standard deviation criteria are used to detect trends in interval and ratio data. The multivariate techniques use a prediction model that predicts the class of both nominal and non-nominal data. These multivariate prototypes include Bayesian, Centroid, Discriminant, and forward-feed neural net models. Data generators were also developed so that each of these models could be compared using different types of problem data. This effort has led to some of the recognizer scripts discussed below.

Overview of Perl DataTools

Much has been said about the value of using a toolkit approach to support the maintenance of software [7]. A good toolkit has a consistent "standard" interface, is portable, open, flexible, extensible, and maintainable. Not all toolkits meet these goals but some that come close use a scripting language to help tie things together. Perl is a scripting language that was designed to tie things together. As a toolkit itself, Perl meets many of the above goals and is also free. Perl has a large support community that includes the Comprehensive Perl Archive Network (CPAN) that is a proven reuse resource. In addition to ease of use and maintenance, the main reason to use a toolkit is to leverage technology in order to create something new. The Perl-based automation tools presented here were designed to be used as leveraging tools that can be extended and hopefully evolve into more useful and interesting tools. Most of the tools are simple scripts that can be used together like a language to capture, recognize, and respond to data in an autonomous way. The goal of this toolkit is to create systems that have little or no interaction with humans -- If a system still needs a user interface than automation isn't complete!

The internal data format used by the majority of these tools is a simple tab delimited table, like a database table with variable names in the first record followed by data for each variable in the remaining records in their respective columns. The data is ASCII so that Perl tools like pattern matching can be used whenever required. The reason for this internal data format will become obvious when examples of the toolkit are described but also many other database-like tools can import data in this format. Any external data that is not in this format, such as binary telemetry data, must be formated so that it these tools can be used. Once in this format, all of the tools can be used to process the data down-stream because they all know the "table-speak" internal data format. Down-stream processing is made easy because most of the tools take their data from standard input and display to standard output so that data can be piped through a stream of tools until all the processing required is accomplished.

Perl DataTools may are organized by their general use as follows:

- HTML generators to present data and document applications
- Data generators to generate test data for building recognizers
- Filters to transform and select data
- Description generators to describe the nature of data
- Plotters to display data graphically
- Movers to put data where it needs to be
- Recognizer technology to classify data so it can be acted upon
- Controllers to respond to recognized conditions

HTML Generators

The HTML generators included here are used to generate hyperlinks to source scripts and data so that the author may explain how scripts work by giving examples of their use. As a matter of fact, the best description of these tools is available by going to the WWW site where these tools are distributed and browsing the HTML. Dbe.pl (Document By Example -- pronounced "Debbie"), creates soft links in a subdirectory of all the files and then creates HTML anchor statements that point to these files when mentioned in the original document. For example, a statement in the original document like "usage: dgen.pl 10 hdata > dgen.out" would create hyperlinks to dgen.pl, hdata, and dgen.out so that the user could browse the content of these files in context. Another script called mkdoc.pl is an upper level script that uses dbe.pl to process all the documents in each subdirectory and create index.html files in each. Additional scripts, mktbl.pl and mkcbox.pl generate HTML tables and checkbox buttons from data sources.

Data Generators

Data Generators are used to generate data so that classifier models can be built and tested using simulated data with known characteristics. These scripts are simulator tools that can be used to generate integer or floating point data with constraints on their range of variation. For example, to use the script dgen.pl the user specifies tab delimited data names on a line, followed by a tab delimited list of respective data types, followed by a tab delimited list of ranges:

```
v1    v2    v3    v4    v5
n      i     o     i     i
1 3   0 9   0 9   0 9   0 9
```

In the second line the n means nominal data, i means interval (or ratio) data, and o means ordinal data. Thus, data named v1 is nominal and it varies randomly from 1 to 3. When dgen.pl is used, the user specifies the number of records to be generated and the name of data generation specification file. The scripts that create a time stamp use the Perl module DateCalc.pm which is an ISO certified module that takes care of messy things involved when a new date and time must be calculated from a given date and time and an offset in days, hours, minutes, and seconds. Table 1 lists the current data generators.

Table 1 -- Data Generators

Script	**Description**

Igen.pl	Generate integer data without time stamp
Fgen.pl	Generate floating point data without time stamp
Tigen.pl	Generate integer data with time stamp
Tfgen.pl	Generate floating point data with time stamp
Dgen.pl	Generate floating point and integer data without time stamp
Cid.pl	Generate integer data with pattern for each classification type

Filters

Filters are used to transform data for easier display and interpretation. For example, script recode.pl can be used to recode (translate) data values of a variable to YES if the value is 1 and to NO is the value is 0, otherwise the value can be coded as UNKNOWN. Script compute.pl can be used to create variables that are based on the values of other variables, such as "v99 = v1*v2 + v3". Script select.pl can be used to select variables and records from a table of data just like a database select statement and script kjoin.pl can be used to join tables of data using a key (1st variable in table). An example of using these tools in combination is as follows:

```
Compute.pl "v5 = v1*v2 + v3" data > v5.tbl
Join.pl data v5.tbl | select.pl -v "v4 v5" "v4 > 0" >v4v5.tbl
```

These statements create a new variable called v5 that is a combination of v1, v2, and v3 and then join this variable with the original table but select only variables v4 and v5 where v4 > 0. Yes, you can do this with some databases but it is often hard to "script" this action. Also, because Perl supports regular expression pattern matching you can also do things like:

```
select.pl  "v5 =~ /0.0/" v4v5.tbl.
```

This will return all the records that match the pattern /0.0/. Table 2 lists the current filters.

Table 2 -- Filters

Script	Description
Cdata.pl	Categorize all floating point data into at max N bins
Compute.pl	Compute new data from current data
Delta.pl	Display difference between each data in current record vs. last record
Dummy.pl	Break-down all categorical data into dummy variables (0,1 values)
Fy.pl	Fisher-Yates shuffle the data
Index.pl	Create an index field for a table of data
Interp.pl	Interpolate time-series data
Join.pl	Join 2 or more tables record by respective record
Kjoin.pl	Join 2 or more tables by use of key fields (relational DB)
Mc.pl	Multicolumn display a single column of data
Mkkb.pl	Generate a PIE KB based from data range specification
Nofirst.pl	Delete 1st field in table
Recode.pl	Recode data according to that described in specification file
Ntime.pl	Return next time given a time and a duration in days hours minutes seconds
Sample.pl	Sample the data every N records
Select.pl	Select subsets of variables and/or records based on selection rule
Skip.pl	Skip N records into data
Smooth.pl	Smooth data using N points
T2s.pl	Tab to space filter for better display
Tstamp.pl	Add time stamp to table of data

Description Generators

Description Generators are report generators that summarize some useful qualities about the data. Some of the most useful of these are the frequency counter, freq.pl, that counts the frequency of occurrence of various values of data and xt.pl that cross-tabulates these counts within the categories indicated by the values of the 1st variable in the table. The script edchk.pl allows the user to specify regular expressions that represent valid data. When this edit-checking tool is used it reports the fields and records that do not match the pattern. The script desc.pl generates descriptive statistics for all the data based on the "type of data" indicated in the data generation specification file mentioned above. Many of these tools are used to build models that can be used to identify types or classes of data that require action. Table 3 lists the current description generators.

Table 3 -- Description Generators

Script	**Description**
Cor.pl	Display correlations among variable in table of data
Desc.pl	Display descriptive statistics for all data in table
Edchk.pl	Edit check data in table
Freq.pl	Display frequency counts for data in table
Msig.pl	Display means and sigmas of data in table
Norm.pl	Normalize data in table
Range.pl	Display ranges of data in table
Xt.pl	Cross-tabulate 1st variable with remaining variables in table

Plotters

Plotters are used to graphically display data and thus "picture" attributes of the data that would otherwise be hard to see. The scripts included here are TkPerl-based scripts that allow simple and convenient plots of data. The data can always be imported into tools like Excel to generate fancy plots. Table 4 lists the current plotters.

Table 4 -- Plotters

Script	**Description**
Plot.pl	Display a timeline of data
Cplot.pl	Display a histogram of data
Mon.pl	Display a real-time timeline of data

Movers

Movers include monitoring tools that grab and move data at critical times. Relmon.pl is a monitor script that looks for a message file that tells it that data is ready to be sent through a firewall. When the message file arrives, the script moves the data into a file queue area along with info about where to send the data and then acknowledges the receipt of the message. Then script fileq.pl (also a monitor script) attempts to send the files through the firewall using shell scripts sendfile and sendfiles and keeps trying until successful. The queue of files to be sent keeps growing as more messages arrive and is finally emptied when the firewall is up and the files are successfully transferred to the remote host.

Recognizer Technology

Recognizers are used to identify critical characteristics of data so that appropriate action can be taken. Some of these tools use the output of the description generators as models to classify data. Such is the case with chk.pl which uses the output from desc.pl, bcl.pl which uses the output of xt.pl, and ccl.pl which uses the output of msig.pl and norm.pl.
A major advantage of using statistical-based models, such as these, is that much of the knowledge acquisition phase of building these kinds of recognizers can be automated. The models are built by simply presenting examples of “good” and “bad” data to the description generators. This is similar to training a neural net to recognize various types of data but experience has shown that it is more reliable and much easier to use the appropriate statistical tool. These types of recognizers lend themselves to applications that require fuzzy lookup such as case-based reasoning.

Script Pie.pl is a Perl Inference Engine (PIE) [5] that uses a backward-chaining search strategy and frame-style rule base to recognize classes of data. It is also very easy to automatically generate a PIE knowledge base that represents limit checking rules to check the quality of data. (Note that a simple limit checker can also be made using the select filter with rules that represent limit checks.) However the real power of PIE is its ability to capture the nature of abstraction hierarchies that represent the nature of diagnostic spaces such as spacecraft subsystems.

All of these recognizers can be configured so they run as simple stream-based filters or as real-time socket-based servers under script cs.pl. Cs.pl provides the infrastructure to connect to a raw data server and serve clients who are interested in knowing the state of the data. It is easy to envision other recognizers that could be built from combinations of these and filter scripts, for example, a change-of-value recognizer that uses PIE to specify maximum value changes of data and script delta.pl that displays differences between current and last values. The set of recognizer technologies presented here is meant to be a useful starting point for a more complete set. Table 5 lists the current recognizer technologies.

Table 5 – Recognizer Technology

Script	**Description**
Chk.pl	Use descriptive statistics information and criteria to check quality of data
Bcl.pl	Use Bayesian model to check quality of data
Ccl.pl	Use centroid model to check quality of data
Dcl.pl	Use discriminant model to check quality of data
Ecl.pl	Use edit check patterns to check quality of data
Scl.pl	Use database style select statement to check quality of data
Pie.pl	Use PIE knowledge base to check quality of data
Cs.pl	Use above models and serve (via socket connection) result of quality check
Ccs.pl	Use simple client to display result from above server
Pieks.pl	Use PIE Knowledge Server (PIEKS) to serve clients
Hsmon.pl	TkPerl-based health and safety client for PIEKS
Ppie.pl	WWW proxy interface to PIEKS

Controllers

Controllers are scripts that use recognizers to identify critical situations and then take appropriate action. The simplest of these scripts is ca.pl that loads a condition-action list that contains the name of the states of critical data conditions paired with the names of scripts to be executed in response to these conditions. Once the condition-action list is loaded the script connects to a server to monitor data and responds appropriately. Ape.pl is A Plan Executor (APE) [5] that continuously monitors via PIEKS while it reactively executes a plan. Pmon.pl is a page monitor that uses the Perl module Telnet.pm, Kermit, and a serial port on a workstation to connect to a ROLM phone and send numeric pages to groups of system engineers. Launch.pl is a TkPerl button interface that allows a user to easily assign labeled buttons to their respective application components and thereby control their execution. This set of controllers allows a great deal of flexibility on a continuum from human control to total autonomy. Table 6 lists the current controllers.

Table 6 – Controllers

Ca.pl	Condition-action script to execute scripts when given conditions are found
Ape.pl	A Plan Executor – reactive plan execution while continuously monitoring
Pmon.pl	Numeric page – monitor page messages and send then through a serial port
Launch.pl	Launch applications by pressing buttons on panel

Conclusions

The data movers that send files through a firewall, the controllers that launch applications, and the numeric paging system that sends pages to system engineers have proven extremely useful for CGRO and are also being used to support the X-ray Timing Explorer, and soon the Advanced Composition Explorer missions.

Perl modules from CPAN like the DBI for Oracle, DateCalc.pm, Telnet.pm, and GIFgraph.pm have proven invaluable. An occasional preview of the modules available on CPAN is well worth any Perl developer’s time.

Although the documentation tools used by ROBOTT are useful, the addition of the document-by-example tool would make maintenance easier for both the maintenance document creator and the software maintenance engineer.

The recognizer script pie.pl and the controller ape.pl were created because they support the same kinds of “expert system” style monitor and control as GenSAA and Genie (less the GUI) but they do it all in Perl. Using all Perl tools would result in applications that are much easier to develop and maintain. The other recognizer and controller technologies that support fuzzy pattern matching may be used to support future missions that use a case-based approach to automation.

Many of the scripts that automate the schedule generation were developed ad hoc and tend to be full of the heuristics required to support CGRO. This is unfortunate because some of the techniques used to support this task would probably be useful to other missions. The authors are of the opinion that if these scripts were written with some of the tools available in Perl DataTools that the scripts would be far easier to maintain and perhaps be made more configurable for other missions.

Parts of Perl DataTools have been in use for a number of years and have proven their usefulness and ease of maintenance. For example, a CGRO software engineer said that it was easy to create a version of the launch tool that keeps track of the state of ROBOTT and indicates this by label and color changes in the buttons. This is the kind of use that is a primary goal of Perl DataTools – Present a set of tools that can easily be understood and extended to suit new applications. Because most of these tools have evolved from ideas that resulted from the hands-on experience supporting real missions they should be of great practical value to other missions as well.

Acknowledgements

The authors wish to thank Julie Breed, Barbie Brown, Bill Stoffel, and Gary Alcott (EDOS) for their support of this effort.

References

1. Stoffel, W., and McLean, D., "Tools for Automating Spacecraft Ground Systems: The Intelligent Command and Control (ICC) Approach," Proceedings of the SpaceOps 96, Munich Germany, 1996.
2. McLean, D., Zhang, Y., and Fatig, M., "Automating the Compton Gamma Ray Observatory Ground System: Expert System and WWW Technologies," Proceedings of the World Congress on Expert Systems, Mexico City, Mexico, January 1998.
3. Hughes, P.M., and Luczak, E.C., "GenSAA: A Tool For Advancing Satellite Monitoring With Graphical Expert Systems," Proceedings of the Second International Symposium on Ground Data Systems for Spacecraft Control, Pasadena, California, November 1992.
4. User's Guide for the Generic Inferential Executor (Genie) Pass Script Builder, GSFC, Greenbelt Maryland, Release 1, March 1996.
5. McLean, D., "A Knowledge Server Toolkit: Perl-based Automation Tools," Proceedings of the 13th Annual AIAA/USU Conference on Small Satellites, North Logan, Utah, August 1999.
6. Chariya Peterson, Karl Mueller, and Nigel A. Ziyad, "Automated Trend Analysis for Spacecraft System", AAAI Spring Symposium, Stanford, CA March 21-23, 1999.
7. McLean, D.R., "Maintaining an Expert Planning System: A Software Tools Approach," Institutionalizing Expert Systems: A Short Handbook for Managers, Jay Liebowitz (Ed.), Printice Hall, 1991.